１．ヒストリー

はじめに

あるソフトウェアの機能や適用例を題材とした論文、紹介記事、技術資料等は多いが、開発過程そのものを扱ったものは少ない。少ない多くは、着手段階での構想・計画や、開発成果の社会的普及を目的とした機能追加歴等に関するものであり、様々の失敗経験を含む開発過程を遡って系統的に記録したヒストリーや、完成段階における諸問題を扱った論考は極めて少ない。

筆者は、15 年来、日中韓にまたがる十数名のプログラマと共に開発に関与してきた景観シミュレータの枝分かれバージョンを統合する作業を 2008-09 年度に行った。本章では、その間の開発の経緯と、整理統合及び完成に向けたチャレンジについて報告する。

１－１．国土交通省版・景観シミュレータ開発過程

バブル崩壊後の 1993～96 年度に実施された建設省総合技術開発プロジェクト「美しい景観の創造技術の開発」の１課題として、建築・都市・土木各分野共通のソフトとして建築研究所と土木研究所の共同で開発が着手され、1996 年からフリーウェアとして WEB 公開された、草分け的な景観検討のための三次元 CG システムである。最初の現場投入は 1996 年の、福岡市内の峰花台団地の建替（中層→高層）で、現場にマシンを持ち込んで事業対象者に自由な視点移動による景観シミュレーションを行った例としては国内初である。

開発に先立って、1992 年に、技術状況を把握するための予備調査を実施し、内外の最新技術とその動向を把握した。1992 年に、ロサンゼルス暴動に際して、復興市街地の三次元ＣＧが作成され、デモンストレーションが行われた。当時、約 500 万円のグラフィック・ワークステーションに、約 300 万円のソフトゥエアを乗せた環境の上で、三次元的に再現された市街地の中を自由に移動する作業環境は、従来の駒落としビデオなどにより作製された固定的なアニメーション（視点や移動経路を変更することはできず、設計内容自体も編集不可能）とは一線を画するものであった。将来のＰＣの高性能化が予想される中で、今後の景観評価・設計検討の手段として有効と考えられた。実際に、現在では１０万円未満のハードにフリーウェアを用いることで同等以上の作業環境が実現している。

開発開始当初(1993)の理念は①マルチプラットフォーム、②オープン・ソース、③フリーウェア、④オブジェクト指向であり、開発にあたっては、ユーザーに触れる外観から作り始め、各種機能を次第に奥の方に作り込んでいく順序とした。

VRML 登場以前の、初期のデータ形式と、基本的な操作（ヒューマン・インターフェース）を現在まで全く変えておらず、Windows VISTA に対応した約 46 万ステップの最新版においても、15 年前のデータとの互換を保っている。平成 13 年に、国総研に移管されてからの、ダウンロードサイトのアクセス数は、35,656 程度である。

（１）プロトタイプ段階(Ver.1.00, 1995)

グラフィック・ワークステーション(UNIX)を用いビューワとしての機能に限定した。暫定データ形式を定め、CAD で作成・変換した構造物三次元データをコンバートし、背景写真と合成する。「視点抽出」機能により背景写真を解析し、同じ視点から構造物の透視図を作り位置精度を追求した。構造物のカラー編集機能を用意した。

（２）モデリング機能の萌芽(Ver.2.03, 1997)

WindowsNT3.5、95 をプラットフォームに加えた。ファイル保存機能と、直方体、球、多角錐､型鋼､階段など様々の基本的な立体図形をパラメトリックに生成する外部関数(.exe)によりモデリング機能を提供した。外部関数はユーザーが追加可能とした。更に景観データベースから、樹木や点景等の基本的な部品を選択し配置できるようにした。全三次元で福岡市内の団地建替えの地元説明、三陸国道で高規格道路の内部検討に使用した。韓国農漁村研究院との共同研究を開始した。

（３）市街地自動生成(Ver.2.05, 1999)

都市計画条件を設定して市街地を自動生成する機能、商店等の写真からテクスチャ付き立体を作成する機能、ステレオ空中写真から地形＋市街地をデータベース化する手法(官民共同研究)など、モデリング機能を拡充し実用性を高めた。福島市の区画整理事務所や幕張駅南口の再開発事務所で地元説明に利用して頂きながら、バグ報告や改善要望に迅速に対応するように努めた。メモリリークを解消し、形状の生成・削除を繰り返すモデリングの持続可能性を高め安定化させた。

ビューワとして固定的なデータに対して視点移動の表示を行うだけであれば、外部ファイルを読み込んでメモリ上にデータを構築するだけで事足りる。しかし、モデリング作業の中で削除したデータが完全に処理されていないと､「ごみ」としてメモリ等を浪費する（メモリリーク）。構築に対応する除却の機能が正しく使われていない場合、あるいはそもそもそのような機能が存在しないような場合に対処する必要があった。プログラマが十分と考えて固定長配列で処理している部分が大きなデータにおいて破綻するような場合についても、動的にメモリブロックを割り当てる方法に修正する必要があった。長期間にわたり、都市計画条件に従って市街地の建物を更新することは、基幹部分に対するストレス・テストとしての意味をもった。

（４）データ WEB 配信(Ver.2.07,2001)

WEB サーバーから配信する三次元データを、WEB ブラウザと景観シミュレータを連携させて表示できるようにした。ユーザーが作成・編集した三次元データを意見と共に投稿し、審査員の判定の上公開するサーバー機能を開発した。15 ヶ所のモデル現場で、ステレオ空中写真から作成した現況市街地データの上にまちづくり計画案を乗せ、WEB 公開した。Windows ME、2000 に対応する中で、多くの潜在バグを発見し解決した。

（５）分野毎に目的を特化した機能追加(1998～2004)

ダムや国営公園等での応用に際して、影、高速表示、地形編集等の部分的機能拡張が行なわれた。現場に対応した枝分かれバージョンが生じた。

機能追加のニーズが生じた場合、その時点における最も安定したソースコードに、追加する方法が最も効率が高い。1997～99 年度に、建築研究所において再開発等の現場担当者による使用をサポートするために、機能追加を抑えて、安定性を追求してデバッグを進めた Ver.2.0X 系列が現在の基幹部分となっている。この時期に、並行して土木研究所において、同じ Ver.2.00 のコア部分に機能を豊富に追加した Ver.3.2、Ver.4.0 の系列が、高規格道路やダムなどの土木系現場における調査業務の中で作成された。2001 年度以降も、国総研において、更に公園などの景観検討のための機能が追加された。結果的に、枝分かれバージョンが生じることとなった。

（６）多言語版(2006)

1997 年に、日韓科学技術協力協定の中に、「景観シミュレーション技術の地域開発への応用」が採択され、建設省建築研究所と韓国農業基盤公社農漁村研究院の間で共同研究が開始された。その中でソフトウェアの韓国語への翻訳と、韓国側での開発成果の還元が行われた。この段階での翻訳移植は、ソースコードを翻訳する形で行った。同様の方法によりインドネシア語への翻訳移植も行った。しかし、この方法では、デバッグや機能追加を行う度に、その結果を反映させるために、現地でのソースコード修正と再ビルドを行う必要があった。

2006 年度に、日韓共同研究の中で試みた国際化を発展させ、言語に依存するメニュー、メッセージ、ヘルプ等を全て外部テキスト化し、同じ実行形式で複数言語を使用できる機構を開発した。これは、インドネシアでの、まちづくりワークショップに投入された。

多言語版においては、デバッグ結果などを直ちに各国に反映できるように、実行形式を言語に依存しない１本に取りまとめ、言語に依存するメニュー、ダイアログ、エラーメッセージ、ヘルプ等を全て外部テキスト化し、起動時に必要な言語を選択的・動的にロードする仕組みとした。これにより、新たな言語における使用も、プログラマや現地語での開発環境なしに、翻訳家によるテキストファイルの作成のみにより実現可能となった。これにより、言語に起因するバージョン枝分かれの問題が解決した。

（７）完成を目指した集約作業(2007～8)

機能が異なる各枝分かれバージョンのソースコードを再整理し、共通の基幹部分を一本化するとともに、分野固有の目的機能を、必要となった時点でロードするプラグイン DLL として、ソースコードにおいて分離・独立させ、選択的に追加できる機構を追加した。その上で、これまでの枝分かれバージョンから、地形編集、トンネル、園路、道路法面を、プラグイン DLL の実装例として分離独立させた。基幹部分に関しては、多言語版とした。

これと共に、プラグイン DLL による新たな機能開発や、新たな言語への翻訳移植、あるいは本システム（図形処理アルゴリスム等）を部品として利用し、システム開発を行なおうとする開発者・プログラマのための解説書を執筆した(本章２～１５章及び付録 A,B)。ソースコードに関しても、整理・整形した上で、本書に CD-ROM として添付した。

図１は、以上の経過を整理したものである。

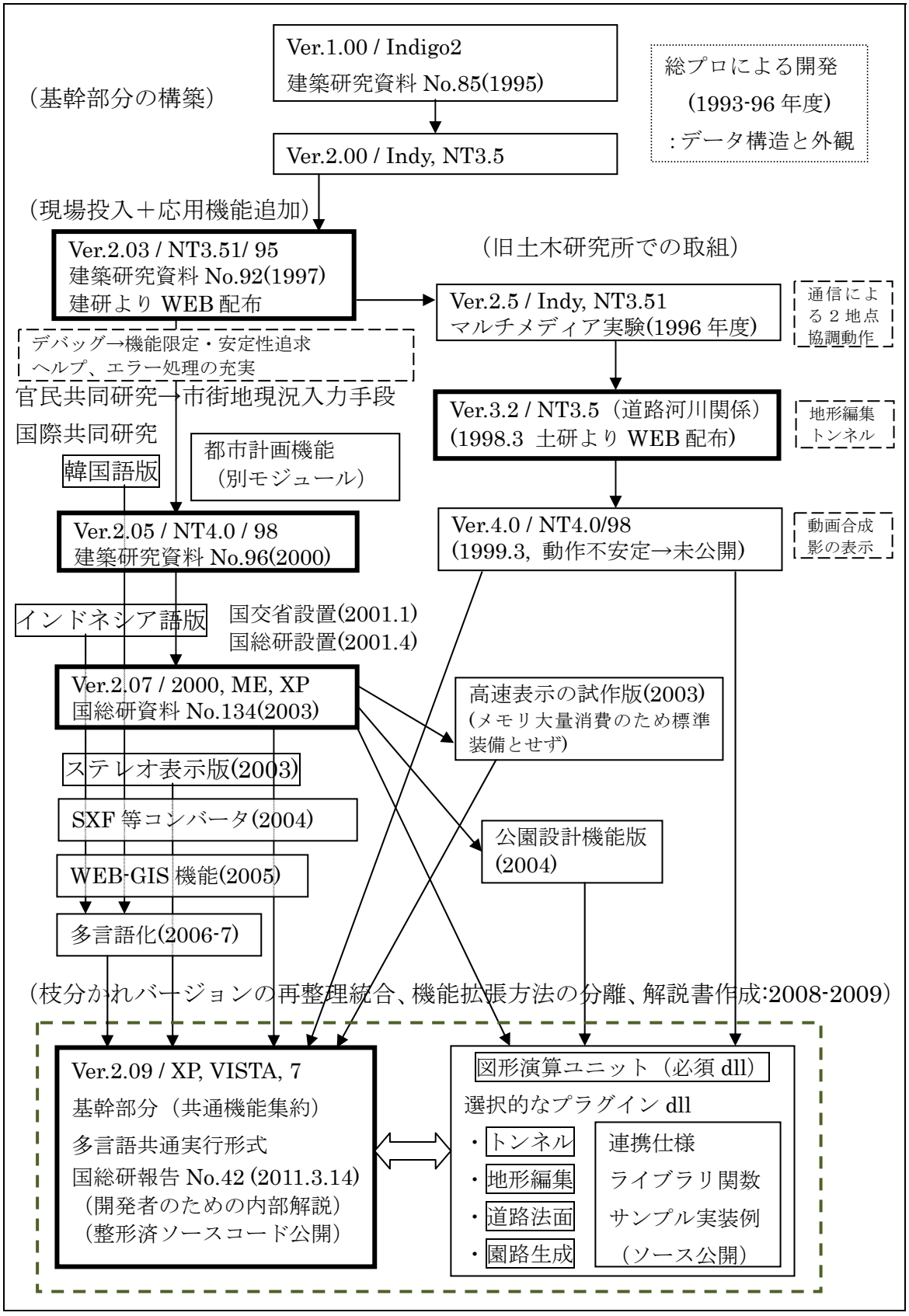

図１－１：景観シミュレータの開発経緯と統合化

１－２．開発における理念と実現過程

1993 年に開発に着手した当初の理念は、「マルチプラットフォーム」「オープン・ソース」「フリーウェア」「オブジェクト指向」である（文献 17 p.282-290）。

（1）マルチプラットフォーム

マルチプラットフォームの考え方は、当時実用的な速度で表示可能なマシンが、UNIX 系のグラフィック・ワークステーションであったこと、それに対して現場で PC の普及が進み、WindowsNT 系の OS にも三次元表示を行うための OpenGL ライブラリが実装可能と予告されていたことによる。実際に、Ver.2.03 以降は、実質的に PC ベースのアプリケーションとなった。しかし、マルチプラットフォームとするために、ソフトウェアの構成を、OS に依存しないデータ処理と、OS に依存するユーザー・インターフェースの部分を分離し、前者をライブラリ関数としたことは、後の Windows 系 OS の頻繁なバージョンアップに対する作業効率に貢献したと考えている。

（2）オープン・ソース

オープン・ソースとした消極的な理由は、限られた開発期間が終了した後に、デバッグやメンテナンスの責任を研究機関が継続的に負担することが困難であることが予想された点にある。非公開としたまま開発と予算措置が終了し、以後適切なサポート体制が組まれない場合、開発担当者の手元に開発成果が死蔵される危険性がある。このことについては、第 16 章で考察する。実行形式だけが公開されているようなシステムに関しては、以後のデバッグや機能追加は非常に困難である。

公開されるオープン・ソースとすることで、特に外注部分のソフトウェア（プログラマの署名が入る）の書法が引き締まったものとなった。但し、その後発展した Linux 等のように、一般のボランティア的な貢献による機能追加やデバッグ等は、これまで行っていない。しかしながら、三次元形状や画像ファイルの処理において、既に公開されていたオープン・ソースのプログラムは、開発担当者プログラマにより適宜引用され、必要に応じてデバッグされている。その中には、Ver.2.09 にむけた統合整理作業の中で、出所が必ずしも明らかにできなかった部分もあることは残念である。しかし、ソースコードの冒頭部分で、担当プログラマによるオリジナルのプログラムであるか否かを明記するようにした。

中国人プログラマに下請けされた一部に、ソースコードではなく、スタティック・ライブラリの形で納品されていた箇所があり、OS のバージョンアップに際して障害の原因となった例がある。結果的にソースコードを入手し解決したが、それまでは、ライブラリのバイナリファイルを修正する必要があった。

オープン・ソースとしたことにより、海外（韓国）との協力が可能となった。WEB 公開されたソースコードの取得・ダウンロードは外国からも可能である。とはいえ、国内予算で開発した知的財産を国際的に無償提供することはフェアではないと考え、協定を締結し

て、役割分担と相手方の開発成果の還元を協力の条件とした。実際に韓国で翻訳移植の作業を共同で実施した所感では、使用する開発環境の違い（それらが含むバグ等）により生じる新たな障害等への対応が付随的に必要となり、単なる言語依存部分の置き換えでは作業は完了せず、ソースコード提供だけでは技術移転は容易ではないという印象を受けた。

Ver.2.07 以前のバージョンにおいて公開したソースコードには、デバッグのためのコンパイル・スイッチや注記、あるいはソースコードの一部を暫定的に削除するためのコメント・アウトなどが多数残されたままの、いわゆるビルドをそのままの形で WEB 公開する方法を採っていた。試行錯誤の記録が残されていることが、将来のデバッグの参考になる可能性もあるが、全般的には、ソースコードを読みにくくしており、このため、これを具体的に理解し改善しようとするプログラマのためには、必ずしも親切ではないと考え、本報告書と共に公開する Ver.2.09 に関しては、不要なコメントを削除した。その際に、手違いによるロジックの変化を防ぐために、リリース・モードでコンパイルした結果生成するオブジェクト・ファイルが、バイナリで変化していないことを確認しながらソースコードの整形作業（外注）を行った。

（3）フリーウェア

国立の研究所が開発したソフトウェアであり、開発に要する費用は国費であることから、開発成果を、メディア代以上の費用で有償販売するような体制は不適当と考えた。

建設省の出先で利用する社内ソフトウェアであれば、必ずしも自由公開する必要はない。しかし、都市計画や景観検討が、国の直轄事業のみならず地方公共団体や公団等においても行われることを考えると、フリーウェアとして扱うことが適当と考えた。また直轄事業の設計は多く民間コンサルタントにより行われているため、民間における使用を制限することは、実用上の支障となる可能性がある。このことから、ランタイム・ライセンスを必要とするようなライブラリを用いて開発手間を節約するような方法を避けた。

また、当時ソフトウェアを配信するために、パソコン通信や、インターネットによるダウンロードが利用可能となっていた。このようは配布方法に際しては、フリーウェアは適切な運用形態である。

このような方法においても、料金を徴収するためには別途技術的な手段を必要とするが、そのような仕組（EC）はまだ発展途上にあった。

ネットワーク上での動作環境を実現した 2001 年度時点(Ver.2.07)では、景観データベースに含まれる事例データや景観データベースのデータがかなりの分量に達していた。このため、その全てをダウンロードすることは、当時の回線速度(10Mbps 程度)では必ずしも実用的ではなかった。そこで、ビューワ（実行形式のみ)、コンパクト（景観構成要素データベースのみ)、フルセット（すべてのデータを含む）の３種類のセットアップを用意し、目的やネットワーク接続環境に応じてダウンロードを選択できるようにすると共に、ネットワークに接続した利用環境において、未取得のデータにアクセスしようとした場合には、

その都度、国土技術政策総合研究所で運用するサーバーから個別に追加ダウンロードするような構成・機構とした。

その後、ネットワークを対象とした悪意のある攻撃などが増加し、それに伴って OS の側でもセキュリティの確保が進んだため、上記のような相互信頼に基づく便利な利用環境の導入にはかなりの慎重さが求められるような環境になった。

（4）オブジェクト指向

オブジェクト指向は、開発開始当時の思潮であった。具体的には C++言語において、クラスを定義し、その下にデータ構造とともに各種オペレーション（関数）もパッケージ化するという手法である。但し、当時まだ C++の開発環境は新しく、コンパイラ等のバグの懸念も払拭されていなかったことから、オブジェクト指向的な考え方を採りつつも、景観シミュレータにおける基本的なデータ形式であるグループや面などのデータに関しては、慣習的な C 言語で、データ構造（構造体定義）とライブラリ関数の作成を一体的に進めた。

具体的なモデリング操作の多くは、まず画面上で処理対象を選択してから処理内容を選択指定する、という作業手順となっており、オブジェクト指向的な考え方に基づくものとなっている。

データ構造においては、「グループ」を基本的な単位として、地物を構造的に記述することとした。具体的に表示される面やそれを定義する頂点座標の「意味あるまとまり」を、一つのグループとして取りまとめる。更に、グループとグループの間にリンクを定義する。これにより、たとえば、親グループとして「テーブル」は、ひとつの「板」子グループと４の「脚」子グループにより構成することができる。データの冗長性を避けコンパクトな記述を可能とするために、同形ならば「脚」グループは一つだけ定義し、これと親グループである「テーブル」の間に４のリンクを定義することもできる。道に沿って多数配列する街灯や、公園に植樹される樹木などは、一つの子グループと多数のリンクにより実現することができる。言いかえると画面に見えているオブジェクトは、「リンク」に対応する。

このグループとして、固定的なデータのみならず、パラメトリックな部品を用意した。形状が固定的なグループであっても、リンクに付属するマトリックスにより XYZ それぞれに独立した倍率をかけることができ、サイズを変更することができる。しかし、土木建築施設を構成する要素の中には、拡大縮小だけでは表現しきれない変形を示す要素がある。例えば、時計は時刻をパラメータとして長針・短針・秒針を変形させるが、その変形は単なる拡大縮小ではない。従って、時刻をパラメータとして、３本の針の配置をコントロールできるような部品が定義できれば、コンパクトに形状を記述することができる。ドアや障子などの可動部品はその類である。また、型鋼のように発注に際していくつかのパラメータを指定することにより最終的な形状が決まる要素も存在している。景観シミュレータにおいては、これらを、外部ファイルの一種として扱い（３－３）、実装においてはユーザーが作成して追加することができる外部関数として実現した（第 5 章）。

（５）データ形式の一貫性と進化

景観シミュレータの開発当初には、三次元データを記述する標準的な形式がまだ存在していなかったことから、独自のデータ形式（内部メモリ上のデータ構造と、外部ファイル形式）を定め、これを用いてシステム開発を進めるとともに、汎用部品や適用事例の蓄積を進めてきた。このデータ形式に関しては、現在まで全く変更を加えていない。

一方、その後 GIS や CAD、仮想現実等における技術が急速に発展する中で、各種データ形式が定義され、一部は ISO によりオーソライズされたこともあるが、以後も様々な形式が提示されている。これらの生成流転に対処して、主要なデータ形式に関する入出力の機能（ファイル・コンバータ）を作成してきた。

景観シミュレータで定めたデータ形式は、地物固有の属性と、具体的な表示のために必要な太陽等の光源配置や、重要な視点などの、個々の情況に関する属性を別ファイルに分離した点に特徴がある。三次元データの WEB 配信を目的とした VRML 言語においては、これらが一つのファイル中に混在している。

地物の表面仕上げ属性に関しては、色彩や鏡面反射率、テクスチャ等を固定的に指定する方法に加えて、表面仕上げ等の表示処理が将来高度化することに備えて、より抽象化した「マテリアル」（木材、鋼材、アスファルト、コンクリート等）を地物に対して間接的に指定しておき、マテリアル毎の具体的な表示方法（テクスチャ、色彩等）を別途詳細指定する方法を用意した。具体的には、材料毎の発光体としての輝度（EMISSION）や、平滑面の鏡面反射率（SPECULAR）等の属性をマテリアル・ファイルの中で定義している。これにより、地物属性を記述する LSS-G ファイル自体の形式の拡張や変更を回避した。

将来、具体的なマテリアルに即して更に高度なグラフィックス表示等が利用可能となった場合、同じ LSS-G ファイルを用いて、それらを利用する可能性が開かれている。

表面から構成される地物の単位である「グループ」を重要な単位として定め、更に一つのグループを、子グループの配置により記述し、その相対的な位置・回転を記述できる。これにより、例えば柱梁などの部材→住宅等の単体→敷地→地区→地域といった段階的な構成を、局地的な座標系と上位の座標系へのリンクとして表現でき、CAD のレイヤー等よりも複雑なデータ構造の記述を可能としている（２－１１参照)。

グループには文字列として任意個数の属性を設定することができ、これまで地面、樹木、住宅などを識別するために使用している。未定義の属性が定義・追加されても、従来のシステムにおいては表示や操作に影響することなく、ファイルの保存・読み込みではその属性は保持・継承される。プラグイン DLL による新機能の開発などにおいて今後積極的に活用される可能性がある（３－３参照)。

形状の大枠を定めておき、形状の一部のみを数値指定する、パラメトリックな部品を外部関数として、標準部品に対して追加が可能な仕組みを用意した。これにより、LSS-G ファイルにおいては、定形的な構成要素をコンパクトに記述することができる。外部関数は

拡張可能である。

以上のような当初の仕組みにより、当初の仕様を変更することなく、その後公開された様々なデータ形式に対しても柔軟に対応することができ、コンバータを追加することにより、データの授受を行ってきた。

Ver.2.09 に向けた整理・統合作業の中で、公開している外部ファイル仕様と、内部の処理ロジックの照合を行い、仕様として定められていながら完全には実装されてかった項目（実際には殆ど利用されていない）に関しても追加で実装を行った。とりわけ、LSS-S 形式における EFFECT の利用に関しては、いくつかの有用な機能を実装している（３－２参照）。

１－３．改良とバージョンの枝分かれ

（1）機能拡張と安定性追求の調和

大規模な機能拡張は、プログラミング業務の役務発注として実施されてきた。新たに追加された機能は、納品段階においては、多くの場合比較的小さなサンプルデータによるテストを経たのみであり、実際の設計対象物等のデータによる検証が不十分である。現場での使用に際して生じるバグは、既に納品を終えた開発チームとは空間的・時間的に隔たっている。最もデバッグが進んだ時期は、1997-8 年であるが、この間のデバッグ、エラー対策、ヘルプの改良等は殆ど研究所職員による直営で行われた。その頃、Windows 版に対応した Ver.2.00 をベースとして、実際の現場でプレゼンテーションを行うと共に Ver.2.05 に向けたデバッグ作業を進める中で、バグの発見を容易にするためにまず、コンパイル警告等を解消した。次に、システム終了時点で生じていた大量のメモリリークを解消するために、不足・欠落していた、使用済みメモリブロックの解放処理を追加した。これにより、バグの探索が容易化された。更に、福島市役所や都市基盤整備公団が実際の土地区画整理事業や市街地再開発事業における現場説明に使用開始したため、現場での実データを用いた操作により生じる障害への対応を直営で行った。この間、デバッグの対象としたビルドに関しては機能追加を行わず、デバッグを中心に作業を進めた。

エラーメッセージに関しても、初期のバージョン 2.03 までは、データの異常や不整合が検出されても可能な限り処理を続行する、という考え方を採っていたが、デバッグの対象としたビルドにおいては、なるべく早期に些細な異常であってもメッセージや警告を表示するようにエラーメッセージを大幅に増補し、現場からのクレームに対して、原因の特定がより容易になるように努めた。

現場での使用と平行して行った、研究所における非常勤職員による景観検討データや景観データベースのコンテンツの拡充に、外部の CAD ではなく開発中のシステム自体を使用したことも、バグの発見に大いに資することとなった。この体制は、Ver.2.07 にいたるまで続けられ、とりわけバグに対して厳しい OS であった Windows ME への移行に際して、更に多くの潜在バグの発見と修正を行うことができた。

同一時期に外注により開発による機能追加が行われたバージョンの枝分かれが生じ

Ver.2.5,3.2,4.0 等に関しては、重大なバグに関しては開発担当者に適宜情報提供を行ったものの、デバッグの結果が必ずしも十分に反映されず、バージョンの枝分かれが生じる結果となった。

一般に、複雑なデータ処理が求められる三次元 CG 製品のデモにおいては、高速視点移動などが重視され、性能の追求に関心が払われる結果、内部の一貫性の確保やバグの除去への時間投入が後手になる状況が生じやすい。民間の CAD 製品と並んで現場説明を行った際に、現場の作業環境で CAD 製品の方が頻繁にシステムダウンを起こすような状況を体験したこともある。とりわけ、発注者やユーザーにとっては、システムの信頼性や安定性の問題は、実務的に本格的に使用する中で初めて認識される価値であるため、外注における納品段階で簡単なサンプルデータを用いたテストとデモにより評価することは難しく、仮に外注を中心とする開発体制であっても、ある程度のインハウスによる継続的サポートは不可欠であった。

長期的に見ると、デバッグを進める時期は、機能追加が停止している期間である方が、全体的な開発効率が高い、と言うことができる。

（2）ハード、OS と開発環境の変化への適応と機能的な進化との歩調合わせの問題

OS と開発環境の発達は、ゼロからのアプリケーション開発を容易化する。従前であれば、開発プロジェクトの中で独自のプログラムを作成しなければならなかった機能の内、各種アプリケーションに共通に利用されるような種類の機能は、OS の機能の一部や、開発環境と共に提供されるサンプル・コードを利用することにより、より迅速・簡便に利用できるようになる。

しかし、一通り完成し、実務に投入されているアプリケーションがそのまま新しい OS で正しく動作するとは限らない。また、機能追加のために、ソースコードをバージョンアップされた開発環境に載せ替えた時に、開発済みの機能がそのまま自動的に正常に動作するとは限らない。これらの環境変化に伴い、全く同じ機能のレベルを維持するだけのために、コストと時間を投入しなければならないことは、メンテナンス上の負担となる。

一方、異なる OS への移行に伴い、潜在バグの顕在化も起こる。とりわけ、2001 年度のネットワーク化に向けた開発において、ユーザー側の OS と想定してテスト・デバッグを行った Windows ME は、既存の多くのソフトウェアで障害が生じることにより、不評・短命な OS であったが、景観シミュレータの場合、OS の欠陥に起因する障害ではなく、それまでの Windows98 上で正常に動作している潜在バグが顕在化した場合が殆どであった。ポインタ変数の不適切な使用や、配列の限界を超えたアクセス、オート変数へのポインタを使い続けることなどによるバグは、それ以前の OS においては、原因発生から暫く操作が行われた後に、顕在化する場合が多く、原因と結果の関係が極めにくい。これに対して、原因発生後直ちに障害となって現れる OS は、むしろ原因の特定を容易化した。このため、その次に長期的にユーザー側の環境となった Windows XP への移行に際しては、殆ど修正を必

要としなかった。Windows VISTA においては、重要な修正は２ヶ所に留まったが、三次元グラフィックスの表示に直接関係する基本的な部分であり、従来の実行形式では全く使用不可能な状況であった（７－１参照)。

Windows 系のサーバー上の機能（WEB-GIS 等）に関しても、OS の 2000Server から 2003Server への移行に際して、「設定されない機能は許可」から「設定されない機能は不許可」に変更がなされ、不正侵入等の原因となる不要なサーバー機能が未設定のまま動作しているような環境が改善された。これに伴い、開発したソフトウェアをセットアップする際の設定作業がより厳格化した（サーバー上で動作するソフトウェアに関しては、別稿に譲ることとし、本書では扱っていない)。

開発環境として C、C++言語を使用した。Ver.1.00 は UNIX 系の C コンパイラを用いたが、Ver.2.00 から、Windows 系の Visual C++を開発環境として併用し、以後こちらが開発の中心的なプラットフォームとなった。開発環境のバージョンは、4.0→6.0→8.0 と推移してきた。2008～9 年度に行った Ver.2.09 に向けた統合作業において開発環境とした 8.0 (VS2005)への移行に際しては、２バイト系の文字列、セキュリティ・ホールとなりうる原始的な C の入出力関数や文字列処理関数等の使用に対する警告が大幅に増補された。これらに伴い、ソースコード全般に対する修正作業が必要となった。

更に、Windows 画面への描画表示に関連する基本的な部分に関しても、OS の変更に伴う仕様変更があり、しばしばソースコードの修正と、最新の OS のみならず旧来の OS 環境上でのテスト・デバッグによる検証を必要とした。

表示速度を向上するためには、様々のノウハウや工夫がある。例えば、バイナリーデータ形式の追加や、OpenGL によるディスプレイ・リストの使用などがある。しかしながら、これらを採用せずとも、開発初期から現在に至るまでの間に、同じ実行形式とデータの組み合わせであっても表示速度は劇的に向上してきた。その最大の要因はコンピュータのハードウェアの速度向上、とりわけ OpenGL 描画処理をハードウェアにより実現する表示部分の機能である。初めて現場に投入した時点で採用した Intergraph 社の TDZ マシンは、WindowsNT3.5 を OS として使用するマシンであったが、ハードウェアの大きな部分を専用グラフィック表示回路が占めていた。その後、CPU 性能に加えて、ゲーム・ソフトの普及に対応した GeForce シリーズ等の、グラフィックス・カードの使用が一般化した。最近では、このような機能が標準でマザーボードに搭載されるようになった。

ディスプレイ・リストを使用する方法は、大容量メモリを必要とするため、2003 年頃に枝分かれバージョンでテストを行ったのみであったが、Ver.2.09 においては大容量メモリの普及状況を考慮して、選択的に使用できるようにした（７－６参照)。

初期から現在までの間に投入した速度向上方法の内、特に有効であったのは、

・ファイル・ロード（IP ライブラリ）におけるシンボル・テーブル検索処理の改善
・ファイル保存時の最適化（同一シンボル名の再利用等）
・地形や周辺市街地のあるデータ表示における三角形と四角形の連続処理

である。

（3）機能追加に際して問題となった事項

個々に行われてきた機能追加に関して、以下のような共通の問題点が指摘できる。

① オペレーションの持続可能性

単純なビューワとしての、言い換えると地物を記述するデータを読み込んで様々な視点から表示するだけの景観シミュレーションであれば、1回のファイル読み込みに伴うデータ構築ができれば十分である。ロードした三次元データを目視確認した後、システムを終了すれば、OS により不要となったリソースは自動的に解放される。しかし、モデリング作業を行う場合、起動から終了までの間に数十～数百回の構築と除却が行われる。更に市街地生成等のシミュレーションを行うためには数万回以上の構築と除却に耐える安定性が求められる。このためには、内部データ構造に対して、構築機能に対応した除却機能や原状復原機能（各種関数）を「可逆的」に作成され、各オペレーションに際してそれらが漏れなく実装されている必要がある。

② 想定外のデータに対する安定性

例えば固定長配列の長さを、プログラマ（必ずしもシミュレーションを行う対象の専門家であるとは限らない）が、「この程度確保しておけば十分」と判断して適当に設定した部分が、データが高度化した後にオーバーフローを起こす。システムのリソース（メモリ容量など）が許す限り、無制限の長さの配列を確保できるようにコーディングするためには、一定の手間が必要となる。まず、コーディングが単純な固定長配列を用いてアルゴリスムの実現を確認し、次に、想定外の大きなデータに備えてメモリブロックの取得と解放処理に書き換える、という手順で作業を進めることが能率的である。メモリブロックを使用する場合、配列長の変更に伴うメモリブロックの再取得(realloc)の処理は一般に処理時間が長いため、配列の長さの変更にある階段を設け、配列の処理がある幅に達した時点で再取得するようにコーディングするが、その幅をどの程度に設定するのが最適かは経験的にしかわからない。

納期が限られた外注の中でこの第二段階をクリアする条件を確保することは難しい。外見で判定できないため、検収段階でソースコードの検査を行い、さらに現場投入後のメンテナンスの中で適宜調整する必要がある。

③ＯＳと開発環境の変化に対する安定性

未来の OS が予見できないため、絶対的な解決はない。同一シリーズのソースコードとそれに基づく実行形式を複数時期の OS 環境上で動作するようにコーディングする努力を、数次にわたる OS の更新にまたがって続ける以外にない。便宜的に OS の種類を認識して条件分岐するようなコーディングは避けるべきである。

機能追加を伴うバージョンアップを外注しようとする場合、ゼロから作り直すことがしばしば提案される。確かに、他者がコーディングしたプログラムを理解した上で、これに

機能を追加することは、ゼロから独自に作成するのとは別の努力を必要とする。景観シミュレーション・システムの場合のように、多くのプログラマによるソースコード作品と作風が共存しているような場合には、最初の読み込みに時間を要するであろう。

しかし、複数の OS や開発環境を越えた安定性を獲得するためには、そのような実績を積み上げるという長い時間が必要である。ゼロからの再開発は、過去に様々な環境で様々なデータを処理してきた、という経験もまたゼロから繰り返す出発点に再び立つ、ということを意味する。

１－４．バージョンの再統合と今後の展望

（1）バージョンの再統合

2008～9 年度に、枝分かれバージョンのソースコードを再確認すると共に、以下のような作業を行った。

①枝分かれバージョンに含まれる共通性の高い基本機能に関しては、基幹部分に追加しデバッグすることにより、有効活用を図った。これには以下のものが含まれる：

・ステレオ表示機能
・影の表示機能
・高速表示機能
・環境設定機能

②枝分かれバージョンに含まれている分野別のニーズに対応した機能については、プラグイン DLL の形で独立させた。

・地形編集（従来、水際形状編集機能として開発されていたもの）
・園路生成
・トンネル

③Ver.2.07 に含まれていた機能の内、改良の余地がある機能に関して、個別にデバッグを進めることは時間を要するため、プラグイン DLL として独立させ、個別のデバッグを将来の課題とした。

・道路法面生成

④多言語機能に関しては、外部関数及びプラグイン DLL も協調動作するように機能拡大を行うと共に、その実装方法をサンプルと共に解説し、サンプルとして配布するこれらのソースコードに実装した。

⑤基幹部分における多言語機能の部分は、コンパイル・スイッチを残しその所在を明示すると共に、コンパイル・スイッチを有効化した（#ifdef MULTI_LANG～#endif）。

⑤将来ニーズの変化により基幹部分に活用可能と考えられる共通機能で、Ver.2.09 に使用しないものに関しては、削除せずにソースコード中に残し、コンパイル・スイッチを用いて無効化した。

・バックアップ・アンドゥ系の機能の内、ユーザー操作をコード化して記録する機能

(#ifdef RDH_NONE～#endif)

（２）機能拡張の方法の明確化と技術資料の整備

短期的に新たなニーズに応えるためには、ソースコードの基幹部分に機能拡張を行う方法が効率的であるが、この方法では無数の枝分かれバージョンが発生する。モデリングのために初期から提供した「外部関数」は、ユーザーが自由に形状生成機能を拡張できるが、現在メモリ上の構築されている地物を削除・編集することができなかった。

2008 年度に導入したプラグイン DLL による方法では、既存の地物に対する編集操作を可能とし、基幹部分の基本的な編集機能と同様にあらゆる地物やシステム状態に対する操作が可能である。そのために、DLL 側から基幹部分のライブラリ関数を利用できる機構とした。

長期的には、開発に関与した人材が入れ替わった後にも、継続的に機能拡張が行える状態を実現することが必要である。このために、本書の中で技術資料を提供した。

（３）記録保存と公開

過去のトラブル対策が個別記録に残されていると、後日、別のトラブルに対処する場合の参考資料となる。景観シミュレータの場合には、1997 年頃から、バグ・レポート様式を用意し１枚紙のレポートに問題と対処の記録を残した。2002 年以降は、情報公開のためのサーバー上にバグ・レポートのコーナー(sim2.nilim.go.jp/AprilFool)を設け、電子的な投稿・記録を行っている。機能改善に関する提案と対処結果についても、同じように処理している。

（４）今後の展望

地理空間情報活用推進基本法(ＮＳＤＩ法)が目指すように、情報化された国土が形成されつつある。１～４次元電子データとしてパーツが蓄積されるとともに、全体マップへの統合が進みつつある。

国土交通省においても、既に運用段階にある電子納品（CALS / EC）において、三次元データが検討され始めた（H20-）。従来は、三次元データ処理は主に機械化施工をターゲットに研究・開発されてきたが、設計から竣工後の維持管理まで利用するとなれば、業務形態を大きく変えることとなる（文献 36)。

景観法はじめ、分野を超えた有形資産を地域の中で全体的に考える機運が生じてきた。低炭素社会等の概念は、可視的な地物を、美的な意味での景観よりも広い概念であるランドスケープとして、地物の背後にある不可視属性の認識を要求している。

①開発の主体

現在とりまとめている構成においては、プラグイン部分を開発することにより、大幅な機能追加を行うことが可能となっている。その主体は任意である。また、運用として、プ

ラグインに関してはオープン・ソースやフリーウェアではない開発方法も可能と考えられる。

②方法

プラグイン部分には、各種機能が追加可能である。現在までの実装例は地形編集等に関係したモデリング目的のものであるが、アニメーション等の表示系の機能も可能である。また、必要であれば、基幹部分よりも巨大なモジュールを開発することも可能である。

一般的に、ソフトウェア開発においては、ゼロベースの開発よりも、ビルドが通り、動作が確認できる既存ソースコードを出発点として機能を作りこんで行く方法が能率的である。プラグイン DLL のサンプルを出発点として開発を進めていくことは、枝分かれの問題を生じることなくローコスト・短期間で開発を進める方法を提供している。

③用途目的

景観をキーワードとする場合、見ることができる有形地物全体の、行政分野を超えた調和を図ることが当面の課題である。たとえ、美しい構造物が建設されても、その周辺にある居住地や里山・田畑等の風景が荒廃すれば、良好な景観は維持できない。このことは、単に風景のレベルにとどまらず、水の流れ、土砂の流出などを通じて、地域の安全性にも影響を及ぼす。更には森林の維持管理の適否が、地球環境にも影響を及ぼしている。

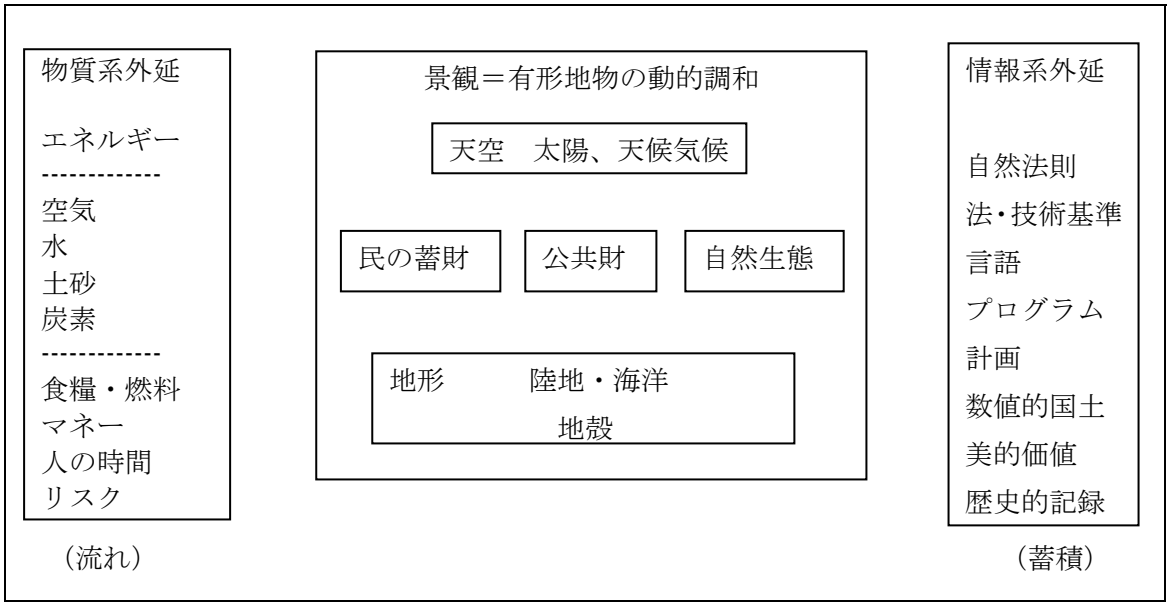

図１－２：景観シミュレーションの対象とその外延

このような事象を広く扱うためには、可視的な地物を扱うのみならず、その背後にある不可視な属性を対象とした評価や数値シミュレーションが必要となる。幸い、国土に関する数値情報やインフラ、市街地等に関する利用可能なデータは標準化され利用可能な形で蓄積され始めており、利活用の可能性が開かれつつある。その一つの方向は、鮮度の高いリアルタイム情報に基づく行政サービスやビジネスモデルであり、もう一つの方向は、歴史的視野に立った、地域の長期的な将来像に関するコンセンサス形成である。

このような目的のためのソフトウェア開発には、それが実現可能なものである限り、今回統合作業を行った結果として提供する Ver.2.09 の基幹部分を、変更することなく、プラグインを開発することにより応用できると考えている。それは、基幹部分よりはるかに大きな規模のものであっても良い。